

# Dimensionality Reduction in Machine Learning

## Taming High-Dimensional Data

Ahmed BADI

Mathematics & Machine Learning Enthusiast

*ahmedbadi905@gmail.com*  
*linkedin.com/in/badi-ahmed*

January 2026

# Outline

- 1 Introduction
- 2 Why Dimensionality Reduction Matters
- 3 Formal Definition
- 4 Linear DR: PCA
- 5 Supervised Linear DR: LDA
- 6 Nonlinear DR: t-SNE and UMAP
- 7 Linear vs Nonlinear Methods
- 8 Applications and Guidelines
- 9 Conclusion

# Introduction

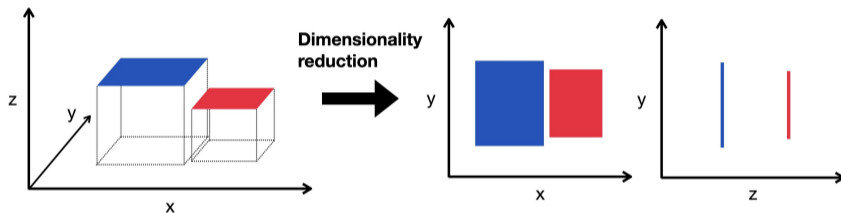
# Motivation

- Many modern datasets have very high dimensionality:
  - Hundreds of features in tabular data.
  - Thousands of pixels in images.
  - Tens of thousands of word counts in text.
- Working directly in such spaces leads to:
  - Higher computational cost.
  - Risk of overfitting.
  - Poor visualization beyond 3D.

# What is Dimensionality Reduction?

- Map data from high-dimensional space  $\mathbb{R}^p$  to lower-dimensional space  $\mathbb{R}^m$  with  $m \ll p$ .
- Goal: preserve the most important structure of the data.
- Closely related concepts:
  - **Feature selection**: choose a subset of existing features.
  - **Feature extraction**: create new features as transformations of the originals.
- Many popular DR methods (PCA, t-SNE, UMAP) are feature extraction techniques.

# High- vs Low-Dimensional View



Illustrative example: projecting 3D data onto a 2D plane while keeping the main structure.

# Why Dimensionality Reduction Matters

# Curse of Dimensionality

- As dimensionality increases, data become sparse and distances behave strangely.
- Many algorithms rely on distance or density:
  - K-Nearest Neighbors.
  - Clustering methods.
- Their performance can degrade severely in high dimensions (“curse of dimensionality”).
- Dimensionality reduction concentrates data into a space where:
  - Distances are more meaningful.
  - Neighborhood-based methods work better.

# Benefits and Trade-offs

## Benefits

- Faster computation: fewer dimensions  $\Rightarrow$  faster training and prediction.
- Noise reduction: discard low-variance or noisy directions.
- Better visualization: 2D/3D embeddings (PCA, t-SNE, UMAP).
- Improved generalization: remove redundant dimensions and reduce overfitting risk.

## Drawbacks

- Some information is inevitably lost.
- Extracted features can be hard to interpret.
- Nonlinear methods can be expensive and sensitive to hyperparameters.

## Formal Definition

# Mapping to a Lower-Dimensional Space

- Given  $\mathbf{x} \in \mathbb{R}^p$ , dimensionality reduction seeks:

$$\phi : \mathbb{R}^p \rightarrow \mathbb{R}^m, \quad \mathbf{z} = \phi(\mathbf{x}), \quad m \ll p.$$

- Desired properties:
  - $\mathbf{z}$  retains useful information (variance, class separability, neighborhood structure, etc.).
  - Resulting representation is usable for tasks like visualization, clustering, classification.

# Linear vs Nonlinear Mappings

- **Linear** methods:

$$z = W^T x, \quad W \in \mathbb{R}^{p \times m}.$$

- Examples: PCA, LDA.
- Easy to compute and interpret.

- **Nonlinear** methods:

- More complex  $\phi$ , e.g., manifold embeddings, neural networks.
- Examples: t-SNE, UMAP, autoencoders.
- Capture curved manifolds in high-dimensional spaces.

# Linear DR: PCA

# Principal Component Analysis (PCA)

- Most widely used linear dimensionality reduction technique.
- Finds orthogonal directions (principal components) that capture maximal variance.
- Applied to centered data  $\tilde{x}_i = x_i - \bar{x}$ .

# Covariance and Eigen-decomposition

- Covariance matrix:

$$C = \frac{1}{n} \sum_{i=1}^n \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^\top.$$

- Solve eigenproblem:

$$C\mathbf{w}_k = \lambda_k \mathbf{w}_k,$$

with  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p \geq 0$ .

- $\mathbf{w}_k$ : principal directions,  $\lambda_k$ : variance along each component.
- First  $m$  components for sample  $\mathbf{x}_i$ :

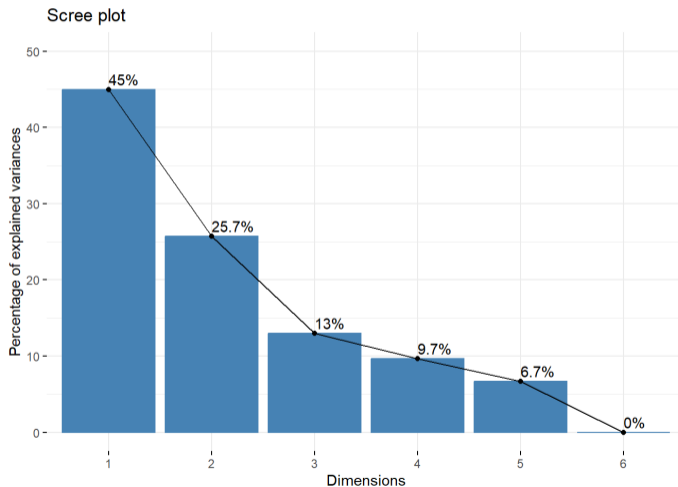
$$\mathbf{z}_i = W^\top \tilde{\mathbf{x}}_i, \quad W = [\mathbf{w}_1, \dots, \mathbf{w}_m].$$

# Variance Explained and Scree Plot

- Fraction of variance explained by first  $m$  components:

$$\text{ExplainedVariance}(m) = \frac{\sum_{k=1}^m \lambda_k}{\sum_{k=1}^p \lambda_k}.$$

- In practice, choose  $m$  such that this ratio  $\geq$  desired threshold (e.g., 90%).



# PCA: Advantages and Limitations

## Advantages

- Simple and computationally efficient.
- Produces decorrelated components.
- Very useful as a preprocessing step and for visualization.

## Limitations

- Linear method: cannot capture complex nonlinear manifolds.
- Components are linear combinations, often hard to interpret.
- Sensitive to scaling: standardization is usually required.

# Supervised Linear DR: LDA

# Linear Discriminant Analysis (LDA)

- Supervised dimensionality reduction method using class labels.
- Seeks projections that maximize between-class separation and minimize within-class spread.
- Especially useful as a preprocessing step for classification.

# Scatter Matrices

- Between-class scatter:

$$S_B = \sum_{k=1}^K N_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^\top.$$

- Within-class scatter:

$$S_W = \sum_{k=1}^K \sum_{i \in \mathcal{C}_k} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top.$$

- $K$ : number of classes,  $N_k$ : samples in class  $k$ ,  $\boldsymbol{\mu}_k$ : class mean,  $\boldsymbol{\mu}$ : global mean.

# LDA Optimization

- LDA maximizes Fisher criterion:

$$J(w) = \frac{w^\top S_B w}{w^\top S_W w}.$$

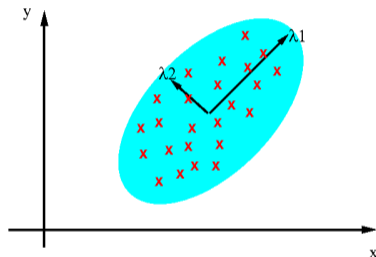
- Solve generalized eigenproblem:

$$S_B w = \lambda S_W w.$$

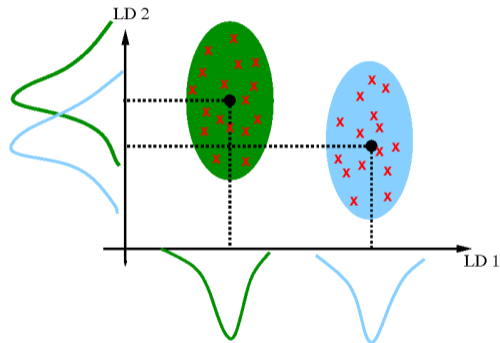
- Discriminant vectors  $w$  define projection directions.
- LDA yields at most  $K - 1$  dimensions for  $K$  classes.

# LDA Projection Example

PCA: component axes that maximize the variance



LDA: maximizing the component axes for class-separation



LDA projection: classes become more separable in the lower-dimensional space.

## Nonlinear DR: t-SNE and UMAP

# Nonlinear Manifold Learning

- Many datasets lie on nonlinear manifolds embedded in high-dimensional spaces.
- Nonlinear dimensionality reduction aims to uncover these manifolds.
- Two popular tools for visualization:
  - t-SNE (t-Distributed Stochastic Neighbor Embedding).
  - UMAP (Uniform Manifold Approximation and Projection).

# t-SNE: Preserving Local Neighborhoods

- In high-dimensional space, define conditional probabilities:

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}.$$

- Symmetrize:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}.$$

- In low-dimensional space ( $y_i$ ):

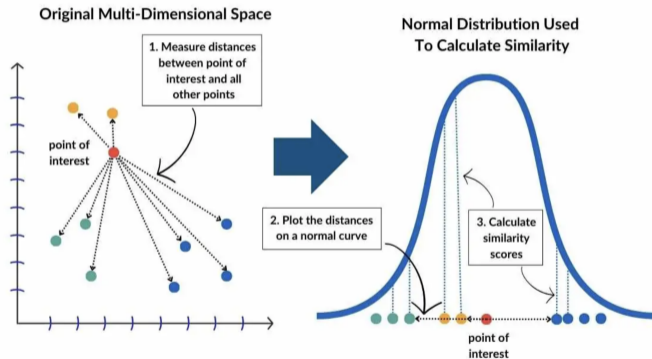
$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq \ell} (1 + \|y_k - y_\ell\|^2)^{-1}}.$$

# t-SNE Objective

- Minimize Kullback–Leibler divergence:

$$KL(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

- Encourages nearby points in high-dimensional space to remain close in low dimensions.
- Reveals clusters and local structure; widely used for visualization.

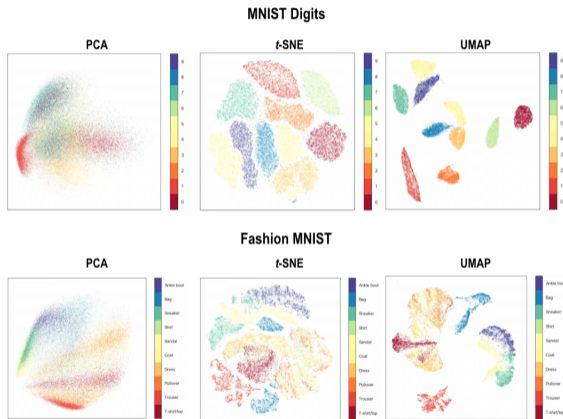


Example of t-SNE embedding into 2D clusters

# UMAP: Graph-Based Manifold Learning

- UMAP models data as a fuzzy topological structure (graph).
- High-level steps:
  - Build a weighted k-nearest-neighbor graph capturing local relationships.
  - View this graph as a fuzzy simplicial set.
  - Learn low-dimensional coordinates by minimizing a cross-entropy loss between high- and low-dimensional fuzzy sets.
- Often preserves both local and some global structure better than t-SNE.
- Typically faster and more scalable.

# UMAP vs t-SNE



Comparison of 2D embeddings (PCA, t-SNE, UMAP) on the same dataset (e.g., MNIST).

# Linear vs Nonlinear Methods

# Linear vs Nonlinear DR

Aspect	Linear (PCA, LDA)	Nonlinear (t-SNE, UMAP, Autoencoders)	Typical Use
Mapping	Linear projection	Complex nonlinear mapping	Visualization, embeddings
Structure	Captures global linear variance	Captures local manifolds and clusters	Complex data (images, text)
Interpretability	Higher	Lower	Depends on need
Compute cost	Lower	Higher	PCA for baseline; t-SNE/UMAP when needed

# Practical Strategy

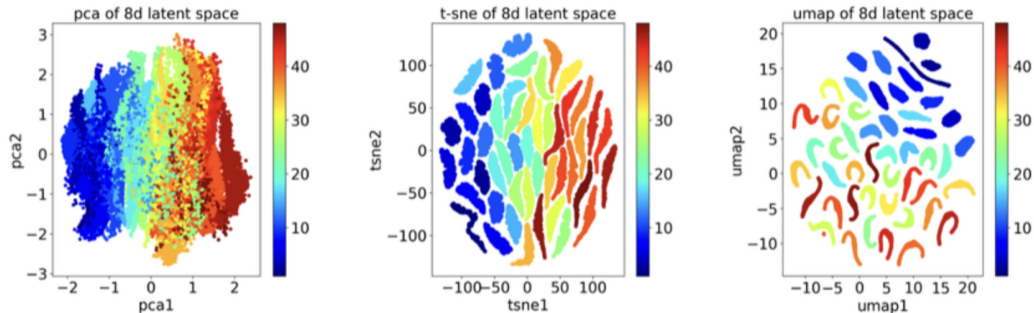
- Start with **PCA**:
  - Fast, gives a strong baseline.
  - Helps choose a reasonable target dimension.
- If patterns remain unclear, use **t-SNE** or **UMAP** for visualization.
- For large-scale or complex tasks, consider **autoencoder-based** DR.

# Applications and Guidelines

# Applications of Dimensionality Reduction

- **Visualization:**
  - Plot word embeddings, latent spaces, or high-dimensional features in 2D/3D.
- **Preprocessing:**
  - Reduce dimensionality before clustering or classification.
- **Noise filtering:**
  - Remove low-variance or noisy components.
- **Compression:**
  - Store lower-dimensional representations of images or signals.

# Latent Space Visualization



Latent space visualization using PCA, t-SNE, and UMAP.

# Practical Guidelines

- **Scale features:**
  - Standardize (mean 0, variance 1) for PCA and many others.
- **Fit on training data only:**
  - Avoid data leakage by applying learned projections to validation/test sets.
- **Choose dimensionality carefully:**
  - Use variance explained (PCA) or validation performance.
- **Use nonlinear methods mainly for visualization:**
  - t-SNE and UMAP are powerful for exploration, but less straightforward for downstream supervised modeling.

# Conclusion

# Summary

- Dimensionality reduction is essential for high-dimensional data:
  - Mitigates the curse of dimensionality.
  - Improves model performance and efficiency.
  - Reveals patterns and clusters through 2D/3D visualizations.
- We covered:
  - Motivation and formal definition.
  - Linear methods: PCA, LDA.
  - Nonlinear methods: t-SNE, UMAP.
- No universally best method:
  - PCA is a strong baseline.
  - t-SNE and UMAP are powerful for visual exploration of complex datasets.

Questions?